Kilobot: A Low Cost Scalable Robot System for Collective Behaviors

Kilobot: A Low Cost Scalable Robot System for Collective Behaviors

Michael Rubenstein, Christian Ahler, and Radhika Nagpal

***Abstract*— In current robotics research there is a vast body of work on algorithms and control methods for groups of decentralized cooperating robots, called a swarm or collective. These algorithms are generally meant to control collectives of hundreds or even thousands of robots; however, for reasons of cost, time, or complexity, they are generally validated in simulation only, or on a group of a few tens of robots. To address this issue, this paper presents Kilobot, a low-cost robot designed to make testing collective algorithms on hundreds or thousands of robots accessible to robotics researchers. To enable the possibility of large Kilobot collectives where the number of robots is an order of magnitude larger than the largest that exist today, each robot is made with only $14 worth of parts and takes 5 minutes to assemble. Furthermore, the robot design allows a single user to easily operate a large Kilobot collective, such as programming, powering on, and charging all robots, which would be difficult or impossible to do with many existing robotic systems.**

I. Introduction

A large group of decentralized closely cooperating entities, commonly called a collective or swarm, can work together to complete a task that is beyond the capabilities of any of its individuals. Many such examples can be found in nature: army ants and honeybee colonies effectively forage over large areas many kilometers wide; desert ant groups can collectively transport large irregular objects 50 times their collective weight; termite colonies construct mounds meters tall even though individuals are only a few millimeters tall themselves. These examples from nature have inspired long-standing research in collective robotics to achieve the kind of parallelism, robustness and collective capability of these natural systems.

Within robotics, there is a wide range of active research topics that explore algorithms to control these robotic collectives, such as self-assembly [1], [16], [14], collective construction [4], [7], and exploration [6], [18], to name a few. Researchers commonly envision these algorithms to operate on collectives of hundreds [4], thousands [16], [14], or more [1], [12], robots; however, for reasons of cost, time, or complexity, they are generally validated in simulation only [1], [14], or on a group of a few tens of robots or fewer [5], [15]. When using a simulation to validate an algorithm for a collective of robots, it is difficult to accurately model robots' interaction with each other, such as communication and sensing, and with the environment, such as movement and collisions. This modeling difficulty can lead to disparities in algorithm behavior when operating on a simulated collective versus a real robotic collective. Additionally, operating an algorithm designed for a large collective of robots on just a few may hide scaling issues within the algorithm that can only be uncovered in a much larger collective. To better understand and validate both current and future collective control algorithms, it would be useful for these algorithms to be tested on a larger collective of real robots.

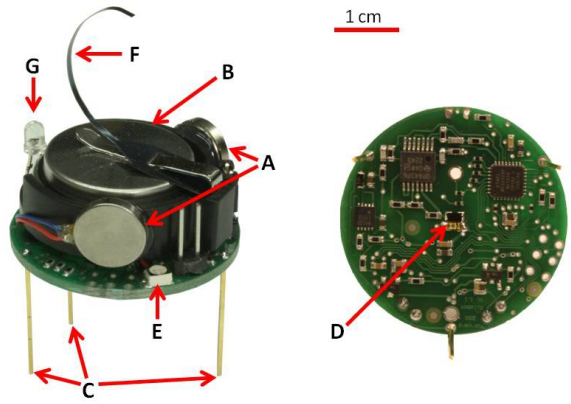

Fig. 1. Isometric (left) and bottom (right) views of a Kilobot. Some key features are: (A) Vibration motors, (B) Lithium-Ion battery, (C) Rigid supporting legs, (D) Infrared transmitter/receiver, (E) Three-color (RGB) LED, (F) Charging tab, and (G) Ambient light sensor. Note the 1cm line for scale.

Other research groups have also recognized the importance of a collective of robots for testing and validating algorithms; however, for various reasons, most operate collectives of a few tens of robots [10], [5], [3] or at the very most a few hundred robots [2], [11]. These collective sizes are primarily limited by robot cost and operational complexity. The cost of an individual robot is perhaps the largest limiting factor for collective size; for a fixed budget, the lower the robot price, the larger the size of the collective. For example, a popular commercially available robot, the e-puck [3], equipped with an infrared communication ring for collective operations, costs over $1300, and as a result, is usually operated in collectives of about 10 robots. A robot more oriented towards large scale multi-robot research is the Jasmine robot [11]. This robot costs about $130 in parts and has been operated in collectives of 150 robots; however, the robot is not available for purchase.

In addition to cost, the complexity of operating the robots, such as turning the robots on/off, charging, controlling, and

Manuscript received September 16, 2011. This work was supported by the Wyss Institute and NSF grant CCF-0829745

Michael Rubenstein is with the School of Engineering and Applied Sciences at Harvard University. mrubenst@seas.harvard.edu

Christian Ahler is with the School of Engineering and Applied Sciences at Harvard University. cahlert@seas.harvard.edu

Radhika Nagpal is with the School of Engineering and Applied Sciences at Harvard University, and the Wyss Institute for Biologically Inspired Engineering. rad@eecs.harvard.edu

programming the collective, also plays a role in limiting the size of the collective. For example, a common way to control the power of a robot is to have a switch on each robot to turn the robot on and off [3], [10]. If this switch takes three seconds per robot to find and flip, then it would take a single person 50 minutes to turn on a collective of 1000 robots! Furthermore, if the collective is large enough, the first robot to turn on may actually run out of power before the last robot is turned on. This manual power switch as well as other design decisions can prevent the size of the collective from scaling to large numbers.

To make a robot scalable to large collective sizes, as described in [8], *all the operations of the robot must work on the collective as a whole*, and not require any individual attention to the robot, such as pushing a switch or plugging in a charging cable for each robot. In other words, all collective operations must be scalable. An example of a scalable operation on a robotic collective is the programming of the I-Swarm robots [13]. In these robots, instead of plugging in a programming cable to each robot in order to update its program, each can receive a program via an infrared communication channel. This allows an overhead infrared transmitter to program all the robots in the collective in a fixed amount of time, independent of the number of robots. Another example of scalable operations is found in [8], [11], where instead of manually plugging in each robot to a charger for battery charging, they use an automatic charging dock that allows the robots to charge themselves without human help, thus making the robot charging scalable. An example of a scalable operation regarding power control is found on the Robomote [15], and in sensor networks. Instead of powering off a robot, it is always on, but in a low power sleep state, ready to turn on if the appropriate command is received. As a result, a Robomote never has to be turned on or off manually, and the entire collective can be turned "on" in a fixed time independent of the number of robots. These sorts of scalable operations are essential for collective operations, but at the same time, they should not dramatically change the robots' capabilities, cost, or ease of manufacturing.

The rest of this paper introduces a new robot, the Kilobot, which is a low cost robot with fully scalable operations. This robot is designed to make testing collective algorithms on hundreds or thousands of robots accessible to robotics researchers. First we describe the hardware design of a Kilobot robot, where its low cost ($14 worth of parts) and quick assembly (5 min) enable large numbers to be produced easily. While these robots are low-cost, they still have abilities similar to other collective robots. These abilities include: differential drive locomotion, on-board computation power, neighbor-to-neighbor communication, and neighbor-to-neighbor distance sensing. These abilities are achieved at low cost mainly through the use of vibration based locomotion and a simple range only sensor. Next, we discuss how the operations of a Kilobot robot, such as programming, turning power on and off, battery charging, and starting/stopping programs, do not require any individual attention by a person, and therefore a large collective can be easily overseen by a single operator.

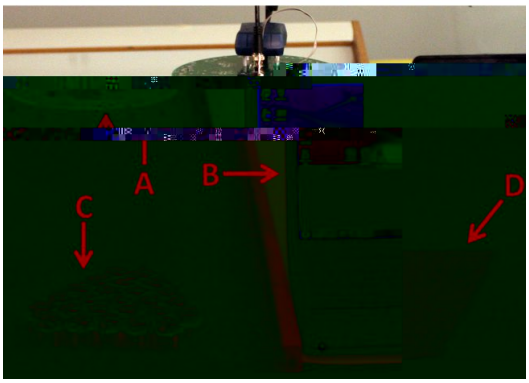

Fig. 2. Picture of the Kilobot arena, including overhead controller (A), control station (B), 25 robot test collective (C), and charging station (D).

II. Kilobot Design

Two competing factors were considered when designing the Kilobot robot: the cost, and the functionality. The robot needs enough functionality to allow it to perform a wide variety of collective behaviors, while at the same time, it must be simple enough to keep the cost low.

SDASH [14], an algorithm developed to self-assemble and self-heal a collective shape, was chosen as a collective behavior to motivate the Kilobot hardware design. This rather complex behavior requires that the robots have the ability to: 1) move forward, 2) rotate, 3) communicate with nearby neighbors, 4) measure distance to nearby neighbors, and 5) have sufficient memory to run SDASH. We feel that these requirements taken from SDASH also give a good sample of robot capabilities needed for many other collective robot behaviors. Furthermore, to improve Kilobots ability to operate in large collectives, as well as to make it a more versitile robotic platform, some additional requirements beyond the five from SDASH were added. The additional requirements are that Kilobot must: measure ambient light levels, display some internal state to assist with debugging, and allow for scalable operations.

While these are not the minimum set of functions needed for a collective robot, they strike a balance between what behaviors a collective of robots is capable of, and the cost of that collective. This section describes the design of the Kilobot hardware which has the functionality desired, while also keeping the price low. A general overview of the Kilobot robot is given in Fig. 1. The environment, called the "arena", that these robots are intended to operate in, consists of a smooth, level, reflective table (e.g. a standard dry erase surface) which can be seen in Fig. 2.

A. Locomotion

One important capability of the Kilobot is that it must be able to move in its environment. The most common locomotion strategy for swarm robots is to use a two-wheeled

differential drive, where each wheel is powered by an electric gear motor. While this conventional wheeled locomotion is quite effective, it is relatively expensive. To keep the cost down, Kilobot uses two sealed coin shaped vibration motors for locomotion. When one of these motors is activated, the centripetal forces generated by the vibrating motor are converted to a forward force on the Kilobot located at the motor's mounting location. The principle of converting the motor vibration to a forward force can be explained using the slip-stick principle, the details of which can be found in [17]. The slip-stick locomotion of a Kilobot was confirmed using high-speed video of the robot's movement. Due to the off-center mounting of the two vibration motors, as shown in Fig. 1, the vibration of one motor alone will cause a rotation of the Kilobot about its vertical axis, while the vibration of the other motor will cause an opposite rotation. By controlling the magnitude of vibration for the two motors independently in a differential drive manner, the robot can move in a continuous range from clockwise rotation, to straight forward, to counterclockwise rotation. This enables the Kilobot to move approximately 1 cm/sec and rotate approximately 45 degrees/sec.

One major drawback to using this low-cost slip-stick based locomotion, as opposed to wheels with encoders, is that there is no real form of odometry. This makes moving precisely over long distances or for a long time difficult. One way to address this difficulty, which harnesses the power of a collective, is to use the measured distances between neighbors as feedback to correct errors in the robot's movement. As is shown in section II-F, this allows the robot to achieve fairly accurate motion control when aided by other robots. Another limitation to this locomotion is that it can not move over rougher surfaces, requiring a smooth surface such as a dry erase surface to work. While this does limit the environments that Kilobot can operate in, it dramatically reduces its cost, and still allows for the demonstration of many interesting collective behaviors.

B. Communication and Sensing

A vast majority of collective robot algorithms use robot-to-robot communication and sensing, such as distance and bearing to neighbors, as the main information to drive the behaviors of individual robots. Therefore, it is critical that Kilobot also be able to communicate with its neighbors and sense some information about its physical relation to its neighbors. In order to keep the robot cost down, the sensing of neighbors only includes distance sensing, not bearing sensing. While bearing sensing is often used with collective robots, for example [9], distance-only sensing is still sufficient for interesting collective behaviors, including SDASH [14].

To communicate with neighboring robots, each Kilobot has an infrared LED transmitter and infrared photodiode receiver, which are located in the center of the PCB and are pointed directly downwards at the table the Kilobot is standing on as shown in Fig. 1. Both the transmitter and receiver have an isotropic emission or reception pattern,

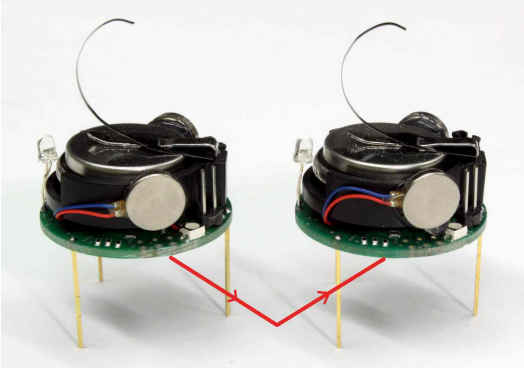

Fig. 3. Illustration showing the reflection path of robot communication.

which allow the robot to receive messages equally from all directions. Additionally, both the receiver and transmitter are wide-angle, with an angle of half power of 60° from the robot's downward pointing vertical axis. When the transmitter is active, any nearby robot can receive the light emitted by the transmitting robot after it is reflected off the table, as shown in Fig. 3. Messages are transmitted by pulsing the transmitter according to standard line coding technique. Using this simple communication method, a Kilobot can communicate at rates up to 30 kb/s with robots up to 10cm (about 6 robot radii) away.

With all robots using the same infrared channel for communication, there is the possibility that two or more robots may try to transmit at the same time. To mitigate this problem a standard carrier sense multiple access with collision avoidance (CSMA/CA) method is used. Even with CSMA/CA, environments with many nearby robots will experience a reduction of the channel bandwidth due to collisions. In an experiment with 25 robots, configured as shown in Fig. 2, the channel could support on average 240 five-byte packets/second, a 32% channel usage.

During any communication between robots, the receiving robot also measures the intensity of the incoming infrared light. This incoming light intensity is a monotonically decreasing function of the distance between the transmitter and the receiver; therefore the distance to the transmitter can be calculated by the receiver. In practice, the incoming intensity of light is also affected by noise and manufacturing variances, which leads to sensing accuracy of ± 2 mm, and precision under 1 mm.

There is also a visible light sensor on each robot, which can sense the level of ambient light shining on the robot. While this sensor is not used in SDASH, it may be useful for other collective applications such as phototaxis or collective transport.

C. Controller

The controller for the robot serves two functions. Firstly, it interfaces with all the low-level electronics such as motors, communication, power circuitry, and the RGB LED (used for displaying information to the operator, seen in Fig. 1). Secondly, it runs a user-defined robot behavior program. The controller used is an Atmega328 microprocessor, which

TABLE I

A SUMMARY OF KILOBOT PARTS COST. PART COSTS ARE BULK PRICES AT QUANTITIES FOR 1000 ROBOTS.

Category	Cost
Locomotion	$3.12
Power	$3.61
Communication/Sensing	$2.20
Control	$2.83
Structure	$1.55
Miscellaneous	$0.74
Total Parts	$14.05

runs at 8 Mhz and has 32K of memory, sufficient space for running an SDASH controller. Some key features of this controller used in the Kilobot are: two pulse width modulation (PWM) channels used for controlling the speed of the vibrating motors, 10-bit analog-to-digital converters used for measuring the incoming infrared light intensity, self-programmable memory used to update the robot's program (described in section III-C), and a low-power sleep mode (see section III-A). The program for the robot is written in C, which allows researchers to quickly and easily develop robot behaviors.

D. Power System

To power the entire robot, each Kilobot has a 3.4 V 160 mAh lithium-ion battery, shown in Fig. 1. This battery can power the robot for 3-24 hours depending on the robot's activity level. Connected to this battery are three voltage regulators and a battery charger. Two of the voltage regulators provide power to the motors and the communication system. Both of these regulators can be switched on and off by the microcontroller, enabling shutdown of the motors and the communication system to conserve power consumption. The third voltage regulator continuously provides power to the microcontroller, and during low-power states (described in section III-A) only draws 30 μA. When the battery charger receives 6 Vdc (described in section III-B), the charger will begin charging the onboard battery; when the battery is charged, the charging will stop.

E. Cost

To allow for large Kilobot collectives, it is critical that each robot be as low-cost as possible. The Kilobot design as described uses about $14 worth of parts, which is at least 10 times less than the lowest cost of currently used collective robots [9]. This cost does not include the assembly of components on the pcb, which can be done by a pick-and-place machine. The cost of each Kilobot can be broken down into six categories: locomotion, power, communication/sensing, control, structure (includes PCB and battery holder), and miscellaneous (all other parts, such as the RGB LED). Table I gives a summary of the cost of these six categories of Kilobot parts.

The assembly time of a robot can also affect the price of a robot if it is pre-assembled; if not, it can make building a collective difficult and time-consuming. Either way, it is important for the robot to be able to be assembled quickly. To aid in a quick assembly time, most of the robot's components are surface-mount, and are placed using a pick-and-place manufacturing robot. The remaining parts to be assembled are: the battery holder, the legs, the motors and the infrared receiver and transmitter. The battery holder and the infrared receiver and transmitter can be assembled by hand. For the remaining parts, custom-made assembly rigs allow for quick and precise alignment and attachment of the motors and the legs. This entire assembly process takes less than five minutes.

F. Robot Capability Demonstrations

To demonstrate Kilobot capabilities, this section shows two demonstrations that provide evidence for the robot's basic functionality: the ability to move within its environment, run a controller, communicate with its neighbors, and measure distance to those neighbors. Additionally, these demonstrations show that when Kilobots sense and communicate with neighbors, they can improve their capabilities beyond what is directly available to them in hardware. In these demonstrations, the improved capabilities shown are position sensing and bearing sensing.

1) Orbit Demonstration: In the first demonstration, a single robot is tasked to travel on a circular path. Lacking odometry, this task is not possible for a single Kilobot, as shown in Fig. 4(C). However, with the assistance of a stationary neighboring Kilobot acting as a marker for the center of the orbit, orbiting is possible, as shown in Fig. 4(A). This stationary robot sends out a message at 1/10th of a second intervals, which is received by the orbiting Kilobot. These messages allow the orbiting Kilobot to compute its current distance to the stationary robot, which is also its distance to the center of its desired orbital path. These distance measurements allow the onboard controller to compute the robot's deviation from the ideal orbit, and using a PD controller, adjust the intensity of both motors' vibrations to correct for it, keeping the robot close to the ideal path.

2) Path Following Demonstration: A second demonstration shows a Kilobot following a more complicated path, in this case a "U" shaped path. This path is defined in Cartesian coordinates as a polygon in which the robot is allowed to move. If the robot is inside the polygon, it moves straight, if not, it turns back towards the interior of the polygon. To enable this behavior, three stationary robots are set in the environment in a triangle shape, as shown in Fig. 4(B). These three robots know their position in a coordinate system and communicate that position to the moving robot 10 times a second. The moving robot uses the communicated positions of the three stationary robots, as well as its measured distance to those robots, to trilaterate its own position in the coordinate system. Once the moving robot knows its position in the coordinate system, it can compute if it is located inside the polygon or not. If not, it can also compute which direction to turn in order to move back into the polygon. The position of the moving robot during five attempts at following the path is shown in Fig. 4(B).

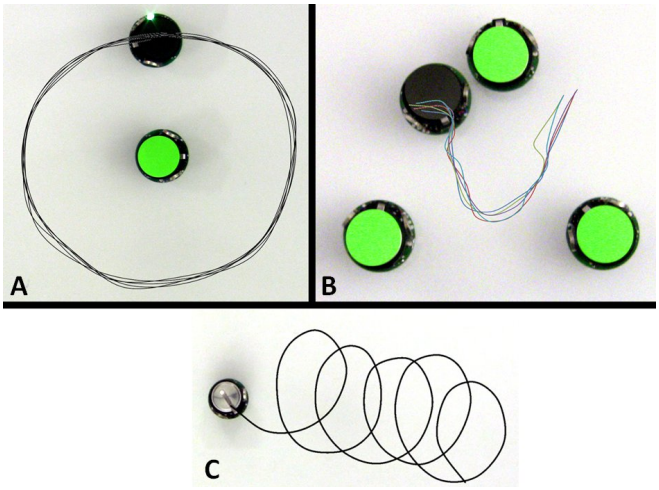

Fig. 4. The path of a Kilobot (black marker) orbiting five times around a second stationary Kilobot (green marker) (A). "U" shaped path following robot (marked with black) and the three stationary beacon robots (marked with green). The path of the moving robot during 5 attempts at following the desired path is shown (B). The path of a single Kilobot attempting to circle without assistance from other robots (C).

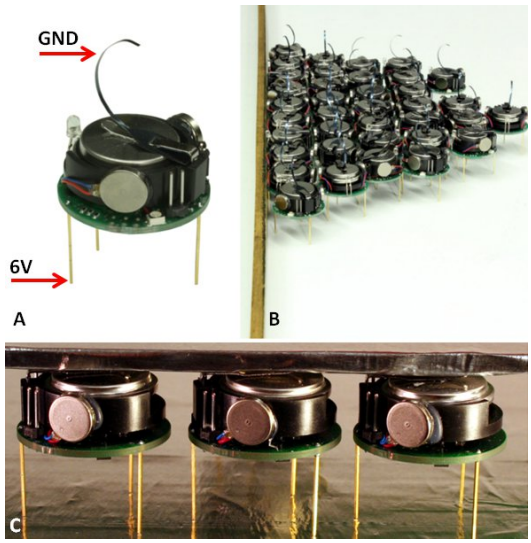

Fig. 5. Connecting to charger input (A); manually pushing robots towards charger (B); a side view of a group of Kilobots charging (C).

III. A Scalable Collective

With a large collective of robots, it can become tedious or even impossible to work with it if the robots require a human operator to interact with each of the robots one at a time. Some of these individual interactions could include pushing the robot's power switch on or off, plugging in a cable to program or charge the robot, and pausing, starting, or stopping the program running on the robot. The following section explains how a Kilobot does not require any individual attention for normal collective operations; i.e., it is a scalable robot. To allow a single user to scalably operate the collective, the setup for Kilobot testing and operations has an overhead infrared controller, as shown in Fig. 2. This controller can send infrared messages, using the same methods as in II-B, to all the Kilobots on the testing arena at once. The overhead controller is in turn controlled by an operator sitting at a computer-based control station.

A. Power

To avoid a physical switch to turn the robots "on" or "off", Kilobots use a power control scheme similar to some sensor networks and the Robomote [15]. This power control scheme works by replacing the standard "off" state of the robot, where the battery is disconnected from the robot, with a low-power sleep mode. While in this mode, the battery stays connected to the robot, but the robot powers down most of its electronics and the microprocessor goes into a sleep state for one minute. During this time, the power drawn from the battery by the Kilobot is approximately 100 μW. After one minute, the microprocessor wakes up, turns on the infrared sensor, and for 10 ms it checks if it receives a wake-up message from the overhead controller. To ensure that a message is received during this 10ms window, the overhead controller transmits the wake-up message every 3ms for over a minute. If a wake-up message is received by the robot, it switches to the standard "on" mode. If no-wake up message is received, then the robot will go back to the low power sleep mode, repeating this cycle until a wake-up message is received. If the robot is in the "on" mode, a sleep message sent from the overhead controller will switch it to the low-power sleep mode.

Using this power control scheme, the robot can remain in the low-power sleep mode, waiting to switch to the "on" mode, for more than 3 months on a single battery charge. Furthermore, the entire collective can be switched from the low power sleep mode to the "on" mode in under one minute, and from the "on" mode to the low-power sleep mode in a few seconds.

B. Charging

As described in section II-D, when 6 Vdc is applied to the input of a Kilobot's battery charger, it will charge the on board battery until the battery is full. To apply this potential to the charger input, the charging tab is connected to an electrical ground, and the bottom legs are connected to 6Vdc, as shown in Fig. 5(A). This enables an entire collective to be charged by first placing it onto a conductive surface, which can easily be done by pushing the whole group with a long stick as shown in Fig. 5(B). Next, a conducting board, such as a poster board coated with metallic tape, is placed on top of the collective. Finally 6Vdc is applied across the bottom conductive surface and the top conducting board, connecting the input of all the chargers to the required voltage. Fig. 5(C) shows a group of Kilobots in this charging configuration which does not require any attention to the individual robots.

C. Programming

To change the program running on the Kilobot's microcontroller, the self-programming feature of the Atmega328 is used. This feature allows the main program code, located in the primary sector of memory, to be overwritten by a program written in a separate "bootloader" sector in memory.

TABLE II

COMPARISON OF SOME COLLECTIVE ROBOT SYSTEMS

Robot	Cost	Scalable Operations	Sensing	Locomotion, speed	body size (cm)	battery life (hours)
Kilobot	$14*	charge, power, program	distance, ambient light	vibration, 1cm/s	3.3	3-24
E-puck[3]	$1,300	none	camera, distance, bearing	wheel, 13cm/s	7.5	1-10
Jasmine[11]	$130*	charging	distance, bearing, light color	wheel, N/A	3	1-2
R-One[9]	$220*	none	visible light, 3d accel, 2d gyro bump, IR sensors, encoders	wheel, 30cm/s	10	6
SwarmBot[8]	N/A**	charge, power, program	Range, bearing,camera, bump	wheel, 50cm/s	12.7	3

* part cost only

** no price information availible

In this bootloader sector of memory, Kilobot has a program that receives infrared messages from the overhead controller which contain portions of the new desired main program code. It then writes these portions of code to the appropriate location in the primary sector of memory. The bootloader program has error-checking to ensure that the program placed in the primary sector is complete and error-free. Once the complete replacement program has been received, the bootloader code resets the microprocessor, and the newly loaded main program code begins execution. When the operator desires to put a new program in all of the Kilobots, the operator transmits a "jump to bootloader" message from the overhead controller. When this message is received by the main program code, it moves the program counter to the bootloader section of code, causing the bootloader program to execute. This scalable programming scheme allows all the Kilobots present in the testing arena to load a new program in under one minute. To validate this, a test collective of 25 robots were programmed using this method in 35 seconds.

D. Other Control

In addition to programming and power control, the overhead controller is also useful in other aspects of Kilobot operations such as querying a robot's battery voltage, and starting, restarting, and pausing the robot's programs. For example, it may be useful for the operator to know the battery voltage of the robots in the collective to determine if they need to be recharged. To do this, the operator sends a message to all the robots via the overhead controller to display voltage. Each robot then displays a color on its RGB LED based on the measured voltage of its battery, displaying green if the battery has more than 90% charge remaining, blue if between 90% and 40% is remaining, and red if less than 40% is remaining. With all robots displaying their charge status, the operator can then look at the collective and determine its overall charge status. In addition, the operator can control the execution of the main program in all Kilobots by issuing commands via the overhead controller to pause, start, stop or restart the main program.

IV. CONCLUSION

In this paper, we have presented the Kilobot robot, designed specifically for operation in a large collective. While the Kilobot is relatively simple compared to other robots as shown in Table II, we believe it capable of running SDASH as well as other collective behaviors. This simplicity allows for low cost, which combined with scalable operations, allow for collectives much larger than are currently available. In the future, we hope to make both the Kilobot hardware designs and software openly available for others to use and to extend. Additionally, we plan on building a 1024 Kilobot collective and implementing SDASH on it.

REFERENCES

[1] D. Arbuckle and A. Requicha. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. Autonomous Robots, 28(2): 197211, 2010.

[2] G. Caprari, P. Balmer, R. Piguet, and R. Siegwart. The autonomous micro robot Alice: a platform for scientific and commercial applications. In Proc. of the Ninth Int. Symp. on Micromechatronics and Human Science, 1998.

[3] F. Mondada, et al. The e-puck, a robot designed for education in engineering. In Proc. of the 9th Conf. on Autonomous Robot Systems and Competitions, 2009.

[4] K. Galloway, R. Jois, and M. Yim. Factory floor: A robotically reconfigurable construction platform. ICRA, 2010.

[5] K. Gilpin, A. Knaian, and D. Rus. Robot pebbles: One centimeter modules for programmable matter through self-disassembly. In ICRA, 2010.

[6] A. Howard, L. Parker, and G. Sukhatme. Experiments with large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. International Journal of Robotics Research, 25(5):431447, 2006.

[7] J. Everist, et al. A system for in-space assembly. IROS, 2004.

[8] J. McLurkin, et al. Speaking swarmish: Human-Robot interface design for large swarms of autonomous mobile robots. In AAAI Spring Symposia, March 2006.

[9] J. McLurkin, et al. A low-cost multi-robot system for research, teaching, and outreach. In DARS, 2010.

[10] M. Jorgensen, E. Ostergaard, and H. Lund. Modular atron: Modules for a self-reconfigurable robot. In IROS, 2004.

[11] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl. Reembodiment of honeybee aggregation behavior in artificial microrobotic system. Adaptive Behavior, 17(3): 237259, 2009.

[12] M. DeRosa, et al. Scalable shape sculpting via hole motion: Motion planning in lattice-constrained modular robots. In ICRA, 2006.

[13] R. Casanova, et al. Enabling swarm behavior in mm^3 sized robots with specific designed integrated electronics. In IROS, 2007.

[14] M. Rubenstein and W. Shen. Automatic scalable size selection for the shape of a distributed robotic collective. In IROS, 2010.

[15] G. Sibley, M. Rahimi, and G. Sukhatme. Robomote: A tiny mobile robot platform for large-scale sensor networks. In ICRA, 2002.

[16] K. Stoy and R. Nagpal. Self-repair through scale independent self-reconfiguration. In IROS, 2004.

[17] P. Vartholomeos and E. Papadopoulos. Analysis, design and control of a planar micro-robot driven by two centripetal-force actuators. In ICRA, 2006.

[18] W. Burgard, et al. Collaborative multi-robot exploration. In ICRA, 2000.